ФЕДЕРАЛЬНОЕ Государственное АВТОНОМНОЕ образовательное УЧРЕЖДЕНИЕ ВЫСШЕГО образования

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ**

**ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»**

**(НИУ «БелГУ»)**

ИНСТИТУТ ИНЖЕНЕРНЫХ И ЦИРОВЫХ ТЕХНОЛОГИЙ

**Кафедра прикладной информатики и информационных технологий**

**Лабораторная работа 3**

**по дисциплине «Разработка программных приложений на языках высокого уровня»**

**студента очной формы обучения**

**направления подготовки 09.03.03. «Прикладная информатика»**

**3 курса группы 12002005**

**Морозова Максима Андреевича**

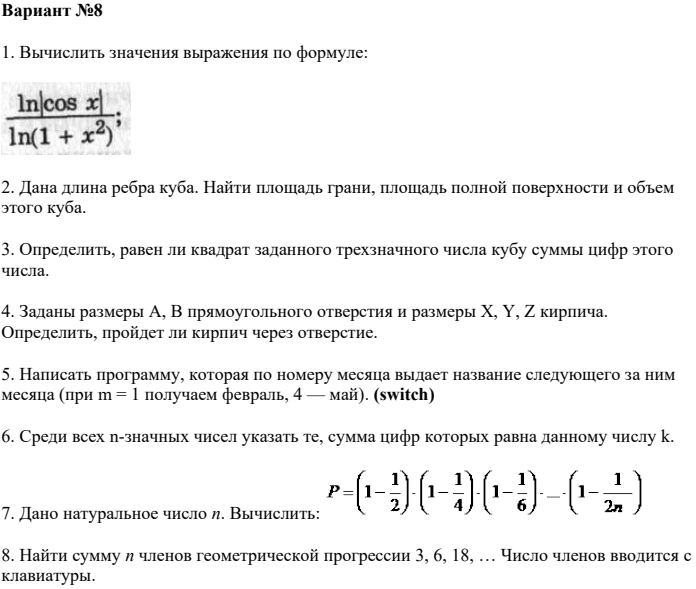
Научный руководитель:

ст. пр. Лифиренко Максим Вячеславович

|  |  |
| --- | --- |
| ***Оценка\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2023 г.  \_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Подпись (расшифровка подписи) |  |

**БЕЛГОРОД, 2023**

Задания выполненные в данной работы:



Результат работы программы 1:



Рисунок 1 – результат работы программы

Результат работы программы 2:

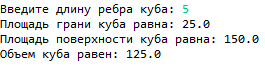


Рисунок 2 – результат работы программы

Результат работы программы 3:



Рисунок 3 – результат работы программы

Результат работы программы 4:

А

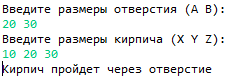


Рисунок 4 – результат работы программы

Код и результат работы задания 5:



Рисунок 5 – результат работы программы

Результат работы программы 6:

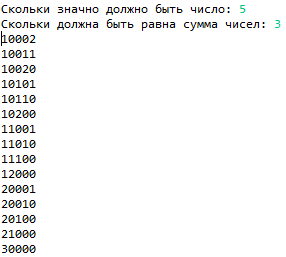


Рисунок 6 – результат работы программы

Результат работы программы 7:



Рисунок 7 – результат работы программы

Результат работы программы 8:



Рисунок 8 – результат работы программы

Листнг программы:

/\*\*

\* @author MaxM

\* @version 11010000.10011111.11010000.10110000.11010000.10111100.11010000.10110000.11010000.10111101.11010000.10111000.11010000.10111101

\*/

package Lab3;

import org.apache.log4j.Logger;

import java.util.Scanner;

public class Brick {

/\*\*

Программа проверяет, пройдет ли кирпич через отверстие заданных размеров.

@param a Размер первой стороны отверстия.

@param b Размер второй стороны отверстия.

@param x Размер первой стороны кирпича.

@param y Размер второй стороны кирпича.

@param z Размер третьей стороны кирпича.

@return Выводит сообщение, указывающее на возможность или невозможность прохождения кирпича через отверстие.

\*/

private static final Logger logger = Logger.getLogger(Brick.class);

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

logger.info("Введите размеры отверстия (A B):");

int a = scanner.nextInt();

int b = scanner.nextInt();

logger.info("Введите размеры кирпича (X Y Z):");

int x = scanner.nextInt();

int y = scanner.nextInt();

int z = scanner.nextInt();

if ((x <= a && y <= b) || (x <= b && y <= a) ||

(y <= a && z <= b) || (y <= b && z <= a) ||

(x <= a && z <= b) || (x <= b && z <= a)) {

logger.info("Кирпич пройдет через отверстие");

} else {

logger.info("Кирпич не пройдет через отверстие");

}

scanner.close();

}

}

package Lab3;

import org.apache.logging.log4j.LogManager;

import org.apache.logging.log4j.Logger;

/\*\*

Класс Cube позволяет вычислить площадь грани, площадь поверхности и объем куба.

\*/

import java.util.Scanner;

public class Cube {

/\*\*

Метод main считывает длину ребра куба от пользователя и вычисляет площадь грани, площадь поверхности и объем куба.

@param args Аргументы командной строки

@return В данной программе не используется.

\*/

private static final Logger logger = LogManager.getLogger(Cube.class);

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Считываем длину ребра куба от пользователя

logger.info("Введите длину ребра куба: ");

double edgeLength = scanner.nextDouble();

// Вычисляем площадь грани, площадь поверхности и объем куба

double faceArea = Math.pow(edgeLength, 2);

double surfaceArea = 6 \* Math.pow(edgeLength, 2);

double volume = Math.pow(edgeLength, 3);

// Выводим результаты вычислений на экран

logger.info("Площадь грани куба равна: " + faceArea);

logger.info("Площадь поверхности куба равна: " + surfaceArea);

logger.info("Объем куба равен: " + volume);

}

}

package Lab3;

/\*\*

Вычисляет сумму первых n членов геометрической прогрессии с первым членом 3 и общим отношением 2.

\*/

import java.util.Scanner;

import org.apache.logging.log4j.LogManager;

import org.apache.logging.log4j.Logger;

public class GeometricProgression {

/\*\*

Входная точка программы.

@param args аргументы командной строки.

\*/

private static final Logger logger = LogManager.getLogger(GeometricProgression.class);

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

logger.info("Введите количество членов геометрической прогрессии: ");

// Считывание количества членов геометрической прогрессии

int n = scanner.nextInt();

// Инициализация первого члена геометрической прогрессии

int firstTerm = 3;

// Инициализация общего отношения геометрической прогрессии

int commonRatio = 2;

// Инициализация переменной суммы

int sum = 0;

// Цикл для вычисления суммы первых n членов геометрической прогрессии

for (int i = 0; i < n; i++) {

// Вычисление i-го члена геометрической прогрессии

int term = firstTerm \* (int) Math.pow(commonRatio, i);

// Добавление i-го члена к сумме

sum += term;

}

// Вывод суммы первых n членов геометрической прогрессии

logger.info("Сумма первых {} членов геометрической прогрессии: {}", n, sum);

}

}

/\*\*

Класс, выполняющий конвертацию номера месяца в название следующего месяца.

\*/

package Lab3;

import org.apache.log4j.Logger;

import java.util.Scanner;

public class Month {

/\*\*

Основной метод программы, выполняющий конвертацию номера месяца в название следующего месяца.

@param args массив аргументов командной строки.

\*/

private static final Logger logger = Logger.getLogger(Month.class);

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

logger.info("Введите номер месяца: ");

int month = input.nextInt();

String nextMonth;

switch (month) {

case 1 -> nextMonth = "февраль";

case 2 -> nextMonth = "март";

case 3 -> nextMonth = "апрель";

case 4 -> nextMonth = "май";

case 5 -> nextMonth = "июнь";

case 6 -> nextMonth = "июль";

case 7 -> nextMonth = "август";

case 8 -> nextMonth = "сентябрь";

case 9 -> nextMonth = "октябрь";

case 10 -> nextMonth = "ноябрь";

case 11 -> nextMonth = "декабрь";

case 12 -> nextMonth = "январь";

default -> {

logger.error("Некорректный номер месяца.");

return;

}

}

logger.info("Следующий месяц после " + month + " - это " + nextMonth + ".");

}

}

/\*\*

Класс, выполняющий конвертацию номера месяца в название следующего месяца.

\*/

package Lab3;

import org.apache.log4j.Logger;

import java.util.Scanner;

public class Month {

/\*\*

Основной метод программы, выполняющий конвертацию номера месяца в название следующего месяца.

@param args массив аргументов командной строки.

\*/

private static final Logger logger = Logger.getLogger(Month.class);

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

logger.info("Введите номер месяца: ");

int month = input.nextInt();

String nextMonth;

switch (month) {

case 1 -> nextMonth = "февраль";

case 2 -> nextMonth = "март";

case 3 -> nextMonth = "апрель";

case 4 -> nextMonth = "май";

case 5 -> nextMonth = "июнь";

case 6 -> nextMonth = "июль";

case 7 -> nextMonth = "август";

case 8 -> nextMonth = "сентябрь";

case 9 -> nextMonth = "октябрь";

case 10 -> nextMonth = "ноябрь";

case 11 -> nextMonth = "декабрь";

case 12 -> nextMonth = "январь";

default -> {

logger.error("Некорректный номер месяца.");

return;

}

}

logger.info("Следующий месяц после " + month + " - это " + nextMonth + ".");

}

}

/\*\*

\* @author MaxM

\* @version 11010000.10011111.11010000.10110000.11010000.10111100.11010000.10110000.11010000.10111101.11010000.10111000.11010000.10111101

\*/

package Lab3;

import org.apache.logging.log4j.LogManager;

import org.apache.logging.log4j.Logger;

import java.util.Scanner;

public class NeturalLogarifm {

/\*\*

\* Метод для вычисления натурального логарифма по заданной формуле

\*

\* @param x входное значение для вычисления

\* @return результат вычисления натурального логарифма по заданной формуле

\*/

private static final Logger logger = LogManager.getLogger(NeturalLogarifm.class);

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

try {

// Вывод приглашения для ввода значения x

System.out.print("Введите x: ");

// Чтение значения x, введенного пользователем, с помощью метода nextDouble() объекта scanner

double x = scanner.nextDouble();

// Вычисление результата выражения, заданного в задаче, и сохранение его в переменную result

double result = Math.log(Math.abs(Math.cos(x))) / Math.log(1 + Math.pow(x, 2));

// Вывод результата вычислений на экран

System.out.println("Ваш результат: " + result);

logger.info("Выполнено успешное вычисление с x = " + x + ", результат: " + result);

} catch (Exception e) {

logger.error("Ошибка при выполнении программы: " + e.getMessage());

} finally {

scanner.close();

}

}

}

/\*\*

Класс Square вычисляет квадрат трехзначного числа и проверяет,

равен ли он кубу суммы цифр этого числа.

\*/

package Lab3;

import org.apache.logging.log4j.LogManager;

import org.apache.logging.log4j.Logger;

import java.util.Scanner;

public class Square {

/\*\*

\* Метод main запрашивает у пользователя трехзначное число, вычисляет

\* сумму его цифр, квадрат исходного числа и куб суммы его цифр.

\* После этого метод сравнивает квадрат числа и куб суммы его цифр,

\* и выводит соответствующее сообщение на экран.

\*

\* @param args массив строковых аргументов командной строки.

\*/

private static final Logger logger = LogManager.getLogger(Square.class);

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

logger.info("Введите трехзначное число: ");

int number = input.nextInt();

int sum = 0; // Инициализация переменной sum (сумма цифр числа)

int currentDigit; // Инициализация переменной currentDigit (текущая цифра числа)

int originalNumber = number; // Сохранение исходного числа в переменной originalNumber

// Цикл while для вычисления суммы цифр числа

while (number > 0) {

currentDigit = number % 10; // Получение текущей цифры числа

sum += currentDigit; // Добавление текущей цифры к сумме

number /= 10; // Уменьшение числа на один разряд

}

// Проверка, что введенное число является трехзначным

if (originalNumber >= 100 && originalNumber <= 999) {

int square = originalNumber \* originalNumber; // Вычисление квадрата исходного числа

int cubeOfSum = (int) Math.pow(sum, 3); // Вычисление куба суммы цифр числа

// Проверка равенства квадрата исходного числа и куба суммы его цифр

if (square == cubeOfSum) {

logger.info("Квадрат числа равен кубу суммы его цифр.");

} else {

logger.info("Квадрат числа НЕ равен кубу суммы его цифр.");

}

} else {

logger.info("Введенное число не является трехзначным.");

}

}

}

/\*\*

\* Класс Summ содержит метод для поиска чисел, соответствующих заданным критериям.

\*/

package Lab3;

import org.apache.log4j.Logger;

public class Summ {

// Создаем объект логгера

/\*\*

\* Метод main() является точкой входа в программу. Он запрашивает у пользователя параметры и вызывает метод findNumbers().

\*

\* @param args аргументы командной строки (не используются)

\*/

private static final Logger logger = Logger.getLogger(Summ.class);

public static void main(String[] args) {

logger.info("Starting program"); // Логируем начало работы программы

Scanner scanner = new Scanner(System.in);

System.out.print("Скольки значно должно быть число: ");

int n = scanner.nextInt();

logger.debug("n = " + n); // Логируем значение n для отладки

// Запрашиваем у пользователя сумму цифр, которую должно составлять число

System.out.print("Скольки должна быть равна сумма чисел: ");

int k = scanner.nextInt();

logger.debug("k = " + k); // Логируем значение k для отладки

scanner.close();

// Находим числа, которые соответствуют заданным критериям

findNumbers(n, k);

logger.info("Program finished"); // Логируем окончание работы программы

}

// Метод для поиска чисел, соответствующих заданным критериям

private static void findNumbers(int n, int k) {

logger.debug("Finding numbers with n = " + n + ", k = " + k); // Логируем параметры для отладки

int[] currentNumber = new int[n];

findNumbersHelper(currentNumber, 0, k); // Вызываем вспомогательный метод

}

// Вспомогательный метод для поиска чисел, соответствующих заданным критериям

private static void findNumbersHelper(int[] currentNumber, int index, int remainingSum) {

// Если все цифры уже выбраны, проверяем сумму цифр

if (index == currentNumber.length) {

if (remainingSum == 0) {

logger.info("Found number: " + arrayToString(currentNumber)); // Логируем найденное число

}

return;

}

// Перебираем все возможные цифры от 0 до 9

for (int i = 0; i <= 9; i++) {

if (i == 0 && index == 0)

{

continue; // Первая цифра не может быть нулем

}

if (remainingSum - i >= 0) {

currentNumber[index] = i; // Записываем цифру в текущее число

findNumbersHelper(currentNumber, index + 1, remainingSum - i); // Рекурсивно вызываем метод для следующей цифры

}

}

}

// Метод для преобразования массива цифр в строку

private static String arrayToString(int[] arr) {

StringBuilder sb = new StringBuilder();

for (int j : arr) {

sb.append(j);

}

return sb.toString();

}

}